# Field-aware Factorization Machines for CTR Prediction

## Abstract

Click-through rate (CTR) prediction plays an important role in computational advertising. Models based on degree-2 polynomial mappings and factorization machines (FMs) are widely used for this task. Recently, a variant of FMs, eldaware factorization machines (FFMs), outperforms existing models in some world-wide CTR-prediction competitions. Based on our experiences in winning two of them, in this paper we establish FFMs as an effective method for classifying large sparse data including those from CTR prediction. First, we propose effcient implementations for training FFMs. Then we comprehensively analyze FFMs and compare this approach with competing models. Experiments show that FFMs are very useful for certain classication problems. Finally, we have released a package of FFMs for public use.

## 1. FM

FM的具体详情可以参考之前的论文总结Factorization Machines 论文阅读总结

### 1.1 FM是什么

将矩阵 $W=w_{i,j}$ 矩阵（这是一个对称方阵）分解成 $W=V^TV$ 的形式，其中 $V=(v_1,v_2,\cdots,v_d)$ 是一个 $k\times d$ 矩阵，且 $k\ll d$，于是 $W$ 矩阵的每一个元素都可以用 $V$ 矩阵对应的两列做内积得到：$w_{ij}=v_i\cdot v_j$，同时多项式模型可以重写，这就是因子分解机模型。

$$y(\mathbf{x})=w_0+\sum_{i=1}^{n}w_ix_i+\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle\mathbf{v}_i,\mathbf{v}_j\rangle x_ix_j$$

由于只需要用分解后产生的 $V$ 就能表达 $W$，使得参数个数由 $d^2$ 变成了 $kd$。另一方面，$V$ 矩阵的每一列 $v_i$ 是第 $i$ 维特征的隐向量，一个隐向量包含 $k$ 个描述第 $i$ 维特征的因子，故称因子分解。

### 1.2 FM能解决参数训练问题的原因

经过因子化之后，组合特征 $x_ix_j$ 和 $x_jx_k$ 的系数 $(v_i\cdot v_j)$ 与 $(v_j\cdot v_k)$ 不再独立，他们共有了 $v_j$，因此所有包含 $x_j$ 特征的非零组合特征的样本都能拿来训练。这是什么意思呢？现在，如果只看交叉项（不管用什么loss，根据链式法则我们总需要乘上 $\frac{\partial f(x)}{\partial w_{ij}}$：

$$f(x)\propto\sum_i\sum_j w_{ij}x_ix_j\rightarrow\frac{\partial f(x)}{\partial w_{ij}}=x_ix_j$$

对于稀疏数据而言，$x_ix_j=0$ 很常见，梯度为0，FM改一下变成：

$$f(x) \propto \sum_{i}^{d} \sum_{j=i+1}^{d} (v_i \cdot v_j) x_i x_j \rightarrow \frac{\partial y}{\partial v_i} = \sum_{j} v_j \cdot x_i x_j$$

原本的多项式模型，为了训练 $w_{ij}$，要求 $x_i$ 和 $x_j$ 不能同时为0，现在我们假设 $x_i \neq 0$，则条件变为“$x_j$ **绝对不可以为0**”。另一方面，同样假设 $x_i \neq 0$，但是对 $j$ 没有限制，在所有的特征中，**任意不为0的** $x_j$ **都可以参与训练**，条件减弱为“**存在** $x_j \neq 0$ **即可**”。因此，**FM缓解了交叉项参数难以训练的问题**。

**总结：** 而FM的提升点主要就是将 $w_{i,j}$ 转换为 $< v_i, v_j >$ 来计算，其中 $v_i, v_j$ 是一个 $k * 1$的矩阵，整个 $V = (v_1, v_2, \cdots, v_d)$ 是一个 $k \times d$ 矩阵，即使不存在特征 $i, j$ 同时非零的样本，依旧可以凭借其他包含组合项计算得到 $v_i, v_j$， 如组合项存在非零值的 $< v_i, v_a >, < v_c, v_j >$等。这才是FM对于稀疏特征处理的最大优势所在，当然，相比于多项式模型，速度提升是第二大优势。

## 1.3 FM计算的复杂度

$$f(x) \propto \sum_{i}^{d} \sum_{j=i+1}^{d} (v_i \cdot v_j) x_i x_j$$

时间复杂度上，若只看交叉项，两层循环 $O(n^2)$，内层k维内积($O(k)$)，综合起来应该是 $O(kd^2)$。然而，交叉项是可以化简的，化简为下面的形式后，复杂度是 $O(kd)$。

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j = \frac{1}{2} \sum_{f=1}^{k} \left( \left( \sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right)$$

## 1.4 FM的梯度下降求解

FM模型方程似乎是通用的，根据任务不同，使用不同的loss。比如，回归问题用MSE，分类问题先取sigmoid或者softmax，然后用cross-entropy，比较灵活。

$$f(x) = w_0 + \sum_{i=1}^{d} w_i \cdot x_i + \frac{1}{2} \sum_{f=1}^{k} \left( \left( \sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right)$$

我们再来看一下FM的训练复杂度，利用SGD（Stochastic Gradient Descent）训练模型。模型各个参数的梯度如下：

$$\frac{\partial}{\partial \theta} y(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

其中，$v_{j,f}$ 是隐向量 $v_j$ 的第 $f$ 个元素。由于 $\sum_{j=1}^{n} v_{j,f} x_j$ 只与 $f$ 有关，而与 $i$ 无关，在每次迭代过程中，只需计算一次所有 $f$ 的 $\sum_{j=1}^{n} v_{j,f} x_j$ 就能够方便地得到所有 $v_{i,f}$ 的梯度。显然，计算所有 $f$的 $\sum_{j=1}^{n} v_{j,f} x_j$ 的复杂度是 $O(kn)$；已知 $\sum_{j=1}^{n} v_{j,f} x_j$ 时，计算每个参数梯度的复杂度是 $O(1)$；得到梯度后，更新每个参数的复杂度是 $O(1)$ 模型参数一共有 $nk + n + 1$ 个。因此，FM参数训练的复杂度也是 $O(kn)$。综上可知，FM可以在线性时间训练和预测，是一种非常高效的模型。

# 2. FFM

## 2.1 FFM (Field-aware Factorization Machine) 原理

FFM（Field-aware Factorization Machine）最初的概念来自Yu-Chin Juan（阮毓钦，毕业于中国台湾大学，现在美国Criteo工作）与其比赛队员，是他们借鉴了来自Michael Jahrer的论文[14]中的field概念提出了FM的升级版模型。通过引入field的概念，FFM把相同性质的特征归于同一个field。以上面的广告分类为例，“Day=26/11/15”、“Day=1/7/14”、“Day=19/2/15”这三个特征都是代表日期的，可以放到同一个field中。同理，商品的末级品类编码生成了550个特征，这550个特征都是说明商品所属的品类，因此它们也可以放到同一个field中。简单来说，同一个categorical特征经过One-Hot编码生成的数值特征都可以放到同一个field，包括用户性别、职业、品类偏好等。在FFM中，每一维特征 $x_i$，针对其它特征的每一种field $f_j$，都会学习一个隐向量 $v_{i,f_j}$。因此，隐向量不仅与特征相关，也与field相关。也就是说，“Day=26/11/15”这个特征与“Country”特征和“Ad_type"特征进行关联的时候使用不同的隐向量，这“Country”和“Ad_type”的内在差异相符，也是FFM中“field-aware”的由来。

假设样本的 $n$ 个特征属于 $f$ 个field，那么FFM的二次项有 $nf$ 个隐向量。而在FM模型中，每一维特征的隐向量只有一个。FM可以看作FFM的特例，是把所有特征都归属到一个field时的FFM模型。根据FFM的field敏感特性，可以导出其模型方程。

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_{i,f_j}, \mathbf{v}_{j,f_i} \rangle x_i x_j$$

其中，$f_j$ 是第 $j$ 个特征所属的field。如果隐向量的长度为 $k$，那么FFM的二次参数有 $nfk$ 个，远多于FM模型的 $nk$ 个。此外，由于隐向量与field相关，FFM二次项并不能够化简，其预测复杂度是 $O(kn^2)$。

此外，内积 $v_{i,f_j} \cdot v_{j,f_i}$，$f_i$ 表示让特征 $i$ 与 特征 $j$ 的 field 关联，同时让特征 $j$ 与 $i$ 的 field 关联，由此可见，FM的交叉是针对特征之间的，而FFM是针对特征与 field 之间的。

> 正是因为FM可以看成FFM中所有特征都属于同一个 field 的特例，故而FM中的每个特征都使用一个隐向量进行表达，而FFM则对每个特征是使用 $f$ 个隐向量表达，因为FFM一共有 $f$ 个field。

另外，值得强调的是，论文中尤其指出，虽然 FFM 相比于 FM，其时间复杂度提升到为 $O(kn^2)$，参数数量提升到 $nfk$ 个，但对每个特征的 $f$ 个 $k$ 长度的隐向量表示，其长度 $k$ 相较于FM有了大幅降低，即文中的：

$$k_{FFM} << k_{FM}$$

即每个特征表达从 1 个长度为$k_{FM}$ 的表达式变成了 $f$ 个长度为 $k_{FFM}$ 的表达式。

## 2.2 实例说明

下面以一个例子简单说明FFM的特征组合方式[9]。输入记录如下

| User | Movie | Genre | Price |
|---|---|---|---|
| YuChin | 3Idiots | Comedy, Drama | $9.99 |

这条记录可以编码成5个特征，其中“Genre=Comedy”和“Genre=Drama”属于同一个field，“Price”是数值型，不用One-Hot编码转换。为了方便说明FFM的样本格式，我们将所有的特征和对应的field映射成整数编号。

| Field name | Field index | Feature name | Feature index |
| --- | --- | --- | --- |
| User | **1** | User=YuChin | **1** |
| Movie | **2** | Movie=3Idiots | **2** |
| Genre | **3** | Genre=Comedy | **3** |
| Price | **4** | Genre=Drama | **4** |
| | | Price | **5** |

那么，FFM的组合特征有10项，如下图所示。

$$\begin{aligned}\langle \mathbf{v}_{1,2}, \mathbf{v}_{2,1} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{1,3}, \mathbf{v}_{3,1} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{1,3}, \mathbf{v}_{4,1} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{1,4}, \mathbf{v}_{5,1} \rangle \cdot 1 \cdot 9.99 \\ + \langle \mathbf{v}_{2,3}, \mathbf{v}_{3,2} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{2,3}, \mathbf{v}_{4,2} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{2,4}, \mathbf{v}_{5,2} \rangle \cdot 1 \cdot 9.99 \\ + \langle \mathbf{v}_{3,3}, \mathbf{v}_{4,3} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{3,4}, \mathbf{v}_{5,3} \rangle \cdot 1 \cdot 9.99 \\ + \langle \mathbf{v}_{4,4}, \mathbf{v}_{5,3} \rangle \cdot 1 \cdot 9.99\end{aligned}$$

其中，红色是field编号，蓝色是特征编号，绿色是此样本的特征取值。二次项的系数是通过与特征field相关的隐向量点积得到的，二次项共有 $\frac{n(n-1)}{2}$ 个。

> 这个公式猛一看显得眼花缭乱，看的时候只关注特征即可，即 $(\mathbf{v}_{i,f_j}, \mathbf{v}_{j,f_i})$ 中的特征 $v$ 下标 $i, j$，公式第一行就是 $i=1, j=2 \cdots 5$，再填补 特征 $v$ 的右下角标 $f_i, f_j$。

# 3. FFM模型优化问题

## 3.1 SGD 优化算法

这里论文中的方法介绍地比较简单，相比之下，美团这篇博客 依据源码总结出FFM的优化步骤相对更具体，此处，摘取美团介绍部分进行分析。

Yu-Chin Juan实现了一个C++版的FFM模型，源码可从Github下载[10]。这个版本的FFM省略了常数项和一次项，模型方程如下。

$$\phi(\mathbf{w}, \mathbf{x}) = \sum_{j_1, j_2 \in \mathcal{C}_2} \langle \mathbf{w}_{j_1, f_2}, \mathbf{w}_{j_2, f_1} \rangle x_{j_1} x_{j_2} \tag{1}$$

其中，$\mathcal{C}_2$ 是非零特征的二元组合，$j_1$ 是特征，属于field $f_1$，$w_{j_1, f_2}$ 是特征 $j_1$ 对field $f_2$ 的隐向量。此FFM模型采用logistic loss作为损失函数，和L2惩罚项，因此只能用于二元分类问题。

$$\min_{\mathbf{w}} \sum_{i=1}^{L} \log\left(1 + \exp\{-y_i \phi(\mathbf{w}, \mathbf{x}_i)\}\right) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

其中，$y_i \in \{-1, 1\}$ 是第 $i$ 个样本的label，$L$ 是训练样本数量，$\lambda$ 是惩罚项系数。模型采用SGD优化，优化流程如下。

参考 Algorithm1, 下面简单解释一下FFM的SGD优化过程。 算法的输入 $tr$、$va$、$pa$ 分别是训练样本集、验证样本集和训练参数设置。

1. 根据样本特征数量（$tr.n$）、field的个数（$tr.m$）和训练参数（$pa$），生成初始化模型，即随机生成模型的参数；
2. 如果归一化参数 $pa.norm$ 为真，计算训练和验证样本的归一化系数，样本 $i$ 的归一化系数为

$$R[i] = \frac{1}{\|\mathbf{X}[i]\|}$$

3. 对每一轮迭代，如果随机更新参数 $pa.rand$ 为真，随机打乱训练样本的顺序；
4. 对每一个训练样本，执行如下操作
    - 计算每一个样本的FFM项，即公式(1)中的输出 $\phi$；
    - 计算每一个样本的训练误差，如算法所示，这里采用的是交叉熵损失函数 $log(1+e\phi)$；
    - 利用单个样本的损失函数计算梯度 $g\phi$，再根据梯度更新模型参数；
5. 对每一个验证样本，计算样本的FFM输出，计算验证误差；
6. 重复步骤3~5，直到迭代结束或验证误差达到最小。

## 3.2 优化技巧

在SGD寻优时，代码采用了一些小技巧，对于提升计算效率是非常有效的。

第一，梯度分步计算。采用SGD训练FFM模型时，只采用单个样本的损失函数来计算模型参数的梯度。

$$\mathcal{L} = \mathcal{L}_{err} + \mathcal{L}_{reg} = \log\left(1 + \exp\{-y_i \phi(\mathbf{w}, \mathbf{x}_i)\}\right) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}_{err}}{\partial \phi} \cdot \frac{\partial \phi}{\partial \mathbf{w}} + \frac{\partial \mathcal{L}_{reg}}{\partial \mathbf{w}}$$

上面的公式表明，$\frac{\partial \mathcal{L}_{err}}{\partial \phi}$ 与具体的模型参数无关。因此，每次更新模型时，只需计算一次，之后直接调用 $\frac{\partial \mathcal{L}_{err}}{\partial \phi}$ 的值即可。对于更新 $nfk$ 个模型参数，这种方式能够极大提升运算效率。

第二，自适应学习率。此版本的FFM实现没有采用常用的指数递减的学习率更新策略，而是利用 $nfk$个浮点数的临时空间，自适应地更新学习率。学习率是参考AdaGrad算法计算的[11]，按如下方式更新

$$w'_{j_1,f_2} = w_{j_1,f_2} - \frac{\eta}{\sqrt{1 + \sum_t (g^t_{w_{j_1,f_2}})^2}} \cdot g_{w_{j_1,f_2}}$$

其中，$w_{j_1,f_2}$ 是特征 $j_1$ 对field $f_2$ 隐向量的一个元素，元素下标未标出；$g_{w_{j_1,f_2}}$ 是损失函数对参数 $w_{j_1,f_2}$ 的梯度；$g^t_{w_{j_1,f_2}}$ 是第 $t$ 次迭代的梯度；$\eta$ 是初始学习率。可以看出，**随着迭代的进行，每个参数的历史梯度会慢慢累加，导致每个参数的学习率逐渐减小**。另外，**每个参数的学习率更新速度是不同的，与其历史梯度有关**，根据AdaGrad的特点，对于样本比较稀疏的特征，学习率高于样本比较密集的特征，因此每个参数既可以比较快速达到最优，也不会导致验证误差出现很大的震荡。

第三，OpenMP多核并行计算。OpenMP是用于共享内存并行系统的多处理器程序设计的编译方案，便于移植和多核扩展[12]。FFM的源码采用了OpenMP的API，对参数训练过程SGD进行了多线程扩展，支持多线程编译。因此，OpenMP技术极大地提高了FFM的训练效率和多核CPU的利用率。在训练模型时，输入的训练参数ns_threads指定了线程数量，一般设定为CPU的核心数，便于完全利用CPU资源。

> 对应论文的3.2小节的共享内存系统的并行化处理

第四，SSE3指令并行编程。SSE3全称为数据流单指令多数据扩展指令集3，是CPU对数据层并行的关键指令，主要用于多媒体和游戏的应用程序中[13]。SSE3指令采用128位的寄存器，同时操作4个单精度浮点数或整数。SSE3指令的功能非常类似于向量运算。例如，aa 和 bb 采用SSE3指令相加（aa 和 bb分别包含4个数据），其功能是 aa 中的4个元素与 bb 中4个元素对应相加，得到4个相加后的值。采用SSE3指令后，向量运算的速度更加快捷，这对包含大量向量运算的FFM模型是非常有利的。

除了上面的技巧之外，FFM的实现中还有很多调优技巧需要探索。例如，代码是按field和特征的编号申请参数空间的，如果选取了非连续或过大的编号，就会造成大量的内存浪费；在每个样本中加入值为1的新特征，相当于引入了因子化的一次项，避免了缺少一次项带来的模型偏差等。

# 4. 增加Field信息

对LIBSVM数据组织形式：

`Lable feat1:val1 feat2:val2 ...,`

其中`(feat, val)`对指示特征索引和值，而对于FFM，将数据组织形式扩展为：

`label filed1:feat1:val1 field2:feat2:val2 ...`

即将相应的域 filed 分配到每个特征。

## 4.1 Categorical Features

对线型模型，分类特征通常转换为多个二进制特征。

`Yes P:ESPN A:Nike G:Male`

`Yes P-ESPN:1 A-Nike:1 G-Male:1`

将每个类别作为一个filed，然后以上数据实例就变为：

`Yes P:P-ESPN:1 A:A-Nike:1 G:G-Male:1`

## 4.2 Numerical Features

考虑以下示例：

| Accepted | AR | Hidx | Cite |
|---|---|---|---|
| Yes | 45.73 | 2 | 3 |
| No | 1.04 | 100 | 50,000 |

表格含义：

- AR: accept rate of the conference
- Hidx: h-index of the author
- Cite: number of citations of the author

一共有两个可能方式分配filed，

1. naive的方式是将每个特征作为dummy filed，生成的数据如下：

   `Yes AR:AR:45.73 Hidx:Hidx:2 Cite:Cite:3`

   然后，dummy filed可能不会增加判定信息, 因为它们只是特征的重复。

2. 另一种方式是离散化数值特征为类别，使用如同对待类别特征的设置添加filed信息，生成的数据如下形式：

   `Yes AR:45:1 Hidex:2:1 Cite:3:1`

   其中AR特征四舍五入为整数。

   **缺点**：这种方式的缺点是难以判定最佳的离散设置，如到底是将45.73设置为45.7、45、40或者甚至是"int(log(45.73))"，此外，离散化可能会损失信息。

## 4.3 Single-filed Features

一些数据集上，所有特征属于单个filed，这样将导致增加filed域毫无意义，这种情况多出来在NLP数据集中。考虑以下样本：

| good mood | sentence |
| --- | --- |
| Yes | Hooray! Our paper is accepted! |
| No | Well, our paper is rejected.. |

这个示例中，唯一的filed就是“sentence”，此时，FFM与FM么什么区别；一般不会为每个单词设置field，因为FFM的时间复杂度为 $O(nkf)$，则使用dummy filed是不现实的，因为此时 $f=n$ 而 特征 $n$ 是非常大的，时间复杂度太高。

# 5. 实验对比

个人对此不展开论述，直接查看原论文更好，只记录几个实验中比较重要的点：

## 5.1 POLY2

论文主要的对比对象为LM, Poly 2 以及 FM，其中LM是线性模型，使用LIBLINEAR实现（一个广泛使用的线型模型工具包）。

Poly 2则是一种近似kernel但又不是kernel的方法，其主要做法是显示地表达出特征的二阶多项式特征映射结果，再套入模型中进行求解，采用的是一种先高维映射，再套入模型求解，最终得到高维映射空间的模型，而kernel方法则是尝试避开显示表达映射后关系，而是采用一种将低纬线性不可分数据映射到高维线性可分，找到高维线性可分分界面再投影回低纬空间得到线型不可分分界超平面的思想。

文中明确指出Poly 2 的映射关系：

$$\phi_{Poly2}(w,x)=\sum_{j_1=1}^{n}\sum_{j_2=j_1+1}^{n}w_{h(j_1,j_2)}x_{j_1}x_{j_2}$$

其中$h(j_1,j_2)$ 是将 $j_1$ 和 $j_2$ 编码为一个自然数的方程，上式计算复杂度是 $O(\overline{n}^2)$，其中 $\overline{n}$是每个实例的非零特征的平均值。

## 5.2 FFM使用指导原则

1. FFM包含大量类别型特征的数据集更有效，需要将类别型特征进行dummy编码（转换成了特征稀疏形式）；
2. 如果转换后的数据集不足够稀疏，则FFM的field域提升效果不明显（极端示例，NLP中常出现的一个文本特征集对应一个filed，此时field起不到作用）；
3. 将FFM适用到数值型特征数据集上没有明显优势（事实上，作者实验验证得到，将FFM适用到全数值特征，当采用dummy filed，即将数值特征当做类别特征进行处理时候，FFM与FM表现近似，filed域起不到作用；而若将数值特征进行离散化处理之后，FFM表现比FM要好，但都差于直接使用dummy filed的效果）。

总结一句话就是**FFM，FM均是针对大规模、特征类、稀疏场景（可通过对类别进行dummy编码得到）具有明显优势。**

其优势一是体现在计算特征组合项的效率上（将$w_{ij}$ 特征组合项分解为只需求解每个特征隐式表达 $< v_i, v_j >$，而$v_i$，$v_j$ 的计算不再要求必须在特征 $i$，与特征 $j$ 同时保持非零时才能更新，而是任意包含 $i$ 特征的非零项均可用于计算$v_i$ 的表达式，$v_j$ 同理；对FM，$v_i$ 使用一个 $k$ 维向量表达，对FFM，$v_i$ 使用一个 $f * k$ 维向量表达）。

# 6. FFM应用

在DSP的场景中，FFM主要用来预估站内的CTR和CVR，即一个用户对一个商品的潜在点击率和点击后的转化率。

CTR和CVR预估模型都是在线下训练，然后用于线上预测。两个模型采用的特征大同小异，主要有三类：用户相关的特征、商品相关的特征、以及用户-商品匹配特征。用户相关的特征包括年龄、性别、职业、兴趣、品类偏好、浏览/购买品类等基本信息，以及用户近期点击量、购买量、消费额等统计信息。商品相关的特征包括所属品类、销量、价格、评分、历史CTR/CVR等信息。用户-商品匹配特征主要有浏览/购买品类匹配、浏览/购买商家匹配、兴趣偏好匹配等几个维度。

为了使用FFM方法，所有的特征必须转换成“field_id:feat_id:value”格式，field_id代表特征所属field的编号，feat_id是特征编号，value是特征的值。数值型的特征比较容易处理，只需分配单独的field编号，如用户评论得分、商品的历史CTR/CVR等。categorical特征需要经过One-Hot编码成数值型，编码产生的所有特征同属于一个field，而特征的值只能是0或1，如用户的性别、年龄段，商品的品类id等。除此之外，还有第三类特征，如用户浏览/购买品类，有多个品类id且用一个数值衡量用户浏览或购买每个品类商品的数量。这类特征按照categorical特征处理，不同的只是特征的值不是0或1，而是代表用户浏览或购买数量的数值。按前述方法得到field_id之后，再对转换后特征顺序编号，得到feat_id，特征的值也可以按照之前的方法获得。

CTR、CVR预估样本的类别是按不同方式获取的。CTR预估的正样本是站内点击的用户-商品记录，负样本是展现但未点击的记录；CVR预估的正样本是站内支付（发生转化）的用户-商品记录，负样本是点击但未支付的记录。构建出样本数据后，采用FFM训练预估模型，并测试模型的性能。

| | #(field) | #(feature) | AUC | Logloss |
|---|---|---|---|---|
| 站内CTR | 39 | 2456 | 0.77 | 0.38 |
| 站内CVR | 67 | 2441 | 0.92 | 0.13 |

由于模型是按天训练的，每天的性能指标可能会有些波动，但变化幅度不是很大。这个表的结果说明，站内CTR/CVR预估模型是非常有效的。

在训练FFM的过程中，有许多小细节值得特别关注。

第一，样本归一化。FFM默认是进行样本数据的归一化，即 pa.normpa.norm 为真；若此参数设置为假，很容易造成数据inf溢出，进而引起梯度计算的nan错误。因此，样本层面的数据是推荐进行归一化的。

第二，特征归一化。CTR/CVR模型采用了多种类型的源特征，包括数值型和categorical类型等。但是，categorical类编码后的特征取值只有0或1，较大的数值型特征会造成样本归一化后categorical类生成特征的值非常小，没有区分性。例如，一条用户-商品记录，用户为“男”性，商品的销量是5000个（假设其它特征的值为零），那么归一化后特征“sex=male”（性别为男）的值略小于0.0002，而“volume”（销量）的值近似为1。特征“sex=male”在这个样本中的作用几乎可以忽略不计，这是相当不合理的。因此，将源数值型特征的值归一化到 [0,1][0,1] 是非常必要的。

第三，省略零值特征。从FFM模型的表达式(1)可以看出，零值特征对模型完全没有贡献。包含零值特征的一次项和组合项均为零，对于训练模型参数或者目标值预估是没有作用的。因此，可以省去零值特征，提高FFM模型训练和预测的速度，这也是稀疏样本采用FFM的显著优势。

# 参考文献

[1] Field-aware Factorization Machines for CTR Prediction [2] 深入FFM原理与实践

[3] Factorization Machines 论文阅读总结

[4] 深入浅出ML之Factorization家族

[5][因子分解机（libffm+xlearn）](https://blog.csdn.net/songbinxu/article/details/79662665)